Pre-Lab Questions

Q1. Which GPIO pin is connected to each of the four LEDs? (Red, Blue, Yellow and Green). Use the schematic not the table as there appears to be a typo in the table. In general, it is a good policy to trust the schematic above other references.

Q2. What are the names and memory locations of the registers you'll need to use to set the GPIO pins associated with the LEDs to be outputs?

Q3. For *each* of the four LEDs indicate which bit number of which register you'll need to write to in order to set that LED's GPIO pin to be an output.

Q4. Assuming the GPIO pins have been setup as outputs, indicate the register names, their addresses and which bits you'll need to write to what value in order to turn on the LEDs.

Q5. Answer the following questions. Assume all values are 16-bit unsigned where bit 15 is the most-significant bit and bit 0 the least significant.

a. What is the value of the number where bit 4 is a 1 and all other bits are zero? Provide your answer in both decimal and hex.
b. What is the value of the number (1<<4)? Provide your answer in both decimal and hex.
c. If x=0x2222, what is the hex value of x after the line **x|=(1<<4)**?
d. Write C code which sets bit 7 and 8 of the variable x.
e. Write C code which clears bit 1 of the variable x using the **&=** operator.
f. Write C code which toggles bit 13 of the variable x using the **^** operator.

Q6. Provide the code below with the blanks correctly filled in.

```
/*
 * main.c
 *       Author: GSI
 */

#include <usbstk5515.h>
#include <stdio.h>

//Addresses of the MMIO for the GPIO out registers: 1,2
#define LED_OUT1 *((ioport volatile Uint16*)__blank a__ )
#define LED_OUT2 *((ioport volatile Uint16*)__blank b__ )
//Addresses of the MMIO for the GPIO direction registers: 1,2
#define LED_DIR1 *((ioport volatile Uint16*)__blank c__ )
#define LED_DIR2 *((ioport volatile Uint16*)__blank d__ )


//Toggles LED specified by index Range 0 to 3
void toggle_LED(int index)
{
        if(index == 3)  //Blue
                LED_OUT1 = LED_OUT1 ^ (1<<(__blank e__));
        else if(index == 2)  //Yellow(ish)
                LED_OUT1 = LED_OUT1 ^ (1<<(__blank f__));
        else if(index == 1)  //Red
                LED_OUT2 = LED_OUT2 ^ (1<<(__blank g__));
        else if(index == 0)   //Green
                LED_OUT2 = LED_OUT2 ^ (1<<(__blank h__));
}

//Makes the GPIO associated with the LEDs the correct direction; turns them off
void My_LED_init()
{
        LED_DIR1  |= __blank i__;
        LED_DIR2  |= __blank j__;

        LED_OUT1  |= __blank k__;  //Set LEDs 0, 1 to off
        LED_OUT2  |= __blank l__;  //Set LEDs 2, 3 to off
}

void main(void)
{
        Uint16 value;
        USBSTK5515_init(); //Initializing the Processor
        My_LED_init();
        while(1)
        {
                printf("Which LED shall we toggle(0, 1, 2, or 3)?\n");
                scanf("%d",&value);
                toggle_LED(value);
        }
}
```

Q7. Consider the zigzag.c code found on the lab website. It draws a zigzag pattern on the screen. *Draw*, freehand, what you expect this code to generate on the LCD. We are particularly concerned with how many zigzags you expect to see. You may find it helpful to read the document SSD1306 128 x 64 Dot Matrix OLED/PLED Segment/Common Driver with Controller (you can find it with a web search). Hint: ignore the OSD9616_send function for now, focus on the state of the **top** and **bottom** arrays. Page 25 of the document will also be helpful. Keep in mind this document refers to a LCD with 8 pages but our LCD has 2 pages.

Q8. Say you have a 50-entry sine table (that is the table[x] has the value for sin(2π*x/50)). Say you output each value for 1ms, then after you do the last value (table[49] in this case) you loop around and output table[0] again. Obviously the sine waves will be a bit choppy…

a. What is the frequency of the sine wave you'd generate?
b. What if you instead held each table entry for two samples in a row (2ms). What would the frequency be?
c. What if you instead drove every other table entry for 1ms (so only table entries 0, 2, 4, etc.). What would the frequency be?
d. Describe what you'd need to do to generate a 1Hz frequency sine wave.
e. Describe what you'd need to do to generate a 15Hz frequency sine wave.

In-lab & Post-lab Questions

Q1. What values do you see at memory locations 0x1c0a as you toggle LED 3 on and off? Does toggling LED1 on and off cause that memory location to change? Why or why not?

Q2. What about memory location 0x1c0b? What values do you see there as you toggle LED 3 and off? Does that value change? Why or why not?

Q3. Rewrite the code for the "**if(index == 3)**" case of **toggle_LED** using these functions rather than the way we did it. What would you say an advantage of using the pointers was rather than this library code? What is an advantage of using the library code rather than the pointers?

Q4. The C5515 has a built-in LCD controller. The folks who designed the C5515 eZDSP Stick chose not to use that built-in controller and instead used an external I2C controller.

a. Explain, in a few sentences, what I2C is.
b. Look at the **OSD9616_send** function. What does it do?
c. In main.c what do you think the purpose of the "top" and "bottom" arrays are? Try to clear (turn off) the entire display. Try to make the entire display turn on.
d. In the prelab you were asked to draw the zigzag figure by hand. Does your answer match what was displayed? What's different if anything?
e. The display is only 16x96 even though the code might lead you to believe it is 16x128. Which parts of top and bottom aren't being displayed?
f. After the printf statement there are a bunch of **OSD9616_send** function calls. What do you suspect their purpose is?

Q5. At what voltage levels does our converter start saturating? (Do not drive more than 8V peak-to-peak to the board please; it can likely handle it, but…). Don't worry about being overly precise (within 0.2V will be fine).

Q6. Go to the C5515_Lib folder and examine the function in usbstk5515_led.c which initializes the ULEDs. Copy that function and describe what is happening line-by-line for that function. You will need to look at the #defines in usbstk5515.h and usbstk5515_led.h file to truly understand what's going on there.

Q7. Go into the AIC_func.c file and provide a line-by-line description of the **AIC_write2** function.